**Claims:**

**1. (Currently amended)** An apparatus comprising:

virtual machine means, instantiated in managed code to execute with a runtime loader, for executing first and second assemblies of one or more files instantiated in the managed code;

means for making a call for access by the first assembly of one or more of the files instantiated in the managed code to the second assembly of one or more of the files instantiated in the managed code; and

means, based upon [[an]] a user identification (ID) for at least one of the first and second assemblies of the one or more files, for determining access privileges of the first assembly of the one or more files to the second assembly of the one or more files.

**2. (Currently amended)** The apparatus as defined in Claim 1, ~~wherein the ID is a user ID~~ further comprising:

an execution engine, instantiated in a native code, to execute the virtual machine in runtime;

a compiler to compile each of the first and second assemblies into native code for execution as native code; and

an operating system in native code to be executed with one or more of the compiled first and second assemblies.

**3. (Previously Presented)** The apparatus as defined in Claim 1, further comprising:

execution engine means, in a native code portion, for executing the virtual machine means in runtime; and

means, in the native code portion, for providing an operating system to be executed with the virtual machine means.

**4. (Previously Presented)** The apparatus as defined in Claim 1, wherein the means for determining access privileges comprises:

means for preventing the access of the first assembly to the second assembly when the determination based upon the ID for at least one of the first and second assemblies is unfavorable based upon predetermined criteria for the respective IDs.

**5. (Previously Presented)** The apparatus as defined in Claim 1, wherein the means for determining access privileges comprises:

means for preventing the access of the first assembly to the second assembly when the ID for the first assembly does not match the ID for the second assembly based upon a predetermined match criteria for the respective IDs.

**6. (Previously Presented)** The apparatus as defined in Claim 1, wherein the means for determining access privileges comprises:

means for preventing the access of the first assembly to the second assembly when the first assembly is in a first application domain and the second assembly is in a second application domain, and the first and second application domains do not match based upon a predetermined match criteria for application domains.

**7. (Previously Presented)** The apparatus as defined in Claim 3, wherein:

the means for determining access privileges comprises means for permitting the access of the first assembly to the second assembly when the ID for the first assembly matches the ID for the second assembly based upon a predetermined match criteria for the respective IDs; and

the apparatus further comprises:

means for compiling at least one of the first and second assemblies from an intermediate language code and metadata into native code;

means for loading the native code with a Common Language Runtime (CLR) loader in the native code portion to load the compiled native code; and

means for executing the compiled native code in the native code portion, wherein the first assembly accesses the second assembly.

**8. (Previously Presented)** The apparatus as defined in Claim 1, wherein the means for determining access privileges comprises:

means for permitting the access of the first assembly to the second assembly when previous access to said second assembly by said first assembly had been permitted.

**9. (Previously Presented** The apparatus as defined in Claim 8, wherein the previous access had been permitted following a prior determination that was favorable based upon a predetermined comparison criteria for the respective IDs.

**10. (Previously Presented)** The apparatus as defined in Claim 1, further comprising:

means for compiling at least one of the first and second assemblies into native code;

verifying means, prior to determining access privileges, for determining whether the ID is accurate for the first and second assemblies;

means, upon the determination by the accuracy means that either of said IDs is inaccurate, for:

permitting the means for compiling to compile at least one of the first and second assemblies into native code; and

delaying the means for determining access privileges until the ID is accurate for the first and second assemblies.

**11. (Previously Presented)** The apparatus as defined in Claim 10, wherein the verifying means is for further determining that the ID is accurate for the first and second assemblies at a runtime for the native code.

**12. (Previously Presented)** The apparatus as defined in Claim 10, wherein the means for delaying the means for determining access privileges is for further halting the delay at a runtime for the native code.

**13. (Original)** The apparatus as defined in Claim 1, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the first and second assemblies in respective application domains.

**14. (Previously Presented)** The apparatus as defined in Claim 3, wherein the execution engine means in the native code portion comprises a compiler to compile each said assembly into native code for execution by the native code portion.

**15. (Previously Presented)** The apparatus as defined in Claim 3, wherein the execution engine means in the native code portion further comprises:

a Just In Time (JIT) compiler to compile each said assembly into native code at runtime; and

a CLR loader to load the compiled native code for execution by the native code portion.

**16. (Original)** The apparatus as defined in Claim 3, further comprising:

means, in the native code portion, for forming a response to the call; and

means for returning the response to the first assembly in the managed code portion.

**17. (Withdrawn)** A method comprising:

calling for a first assembly in a managed code portion to have access to a second assembly in the managed code portion;

preventing the access by the first assembly to the second assembly upon intercepting the call;

verifying, based upon an ID for at least one of the first and second assemblies, that the first assembly is privileged to access the second assembly;

compiling at least one of the first and second assemblies from an intermediate language code and metadata into native code;

loading the native code with a CLR loader in a native code portion that includes an operating system; and

executing the compiled native code in the native code portion, wherein the first assembly accesses the second assembly.

**18. (Withdrawn)** The method as defined in Claim 17, wherein the ID is a user ID.

**19.** **(Withdrawn)** The method as defined in Claim 17, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the first and second assemblies in respective application domains.

**20.** **(Withdrawn)** The method as defined in Claim 17, wherein:

the compiling is performed by a JIT compiler; and

the native code portion includes a CLR loader to load the compiled native code.

**21.** **(Withdrawn)** The method as defined in Claim 17, wherein the verifying verifies that the first assembly is privileged to access the second assembly when the first assembly has previously been privileged to access the second assembly.

**22.** **(Withdrawn)** The method as defined in Claim 21, wherein the first assembly has previously been privileged to access the second assembly following a prior said verifying, based upon the ID for at least one of the first and second assemblies, that the first assembly is privileged to access the second assembly.

**23.** **(Withdrawn)** The method as defined in Claim 17, further comprising:

forming a response to the call in the native code portion; and

returning the response to the first assembly in the managed code portion.

**24.** **(Withdrawn)** A computer readable medium including machine readable instructions for implementing the method as defined in claim 17.

**25.** **(Withdrawn)** A method comprising:

calling for a first assembly in a managed code portion to access to a second assembly in the managed code portion;

determining whether respective IDs associated with the first and second assemblies are accurate;

when the determining determines an inaccuracy:

compiling at least one of the first and second assemblies from an intermediate language code and metadata into native code; and

when the respective IDs associated with the first and second assemblies are accurate:

verifying, based upon the ID for at least one of the first and second assemblies, whether the first assembly is privileged to access the second assembly.

**26.** **(Withdrawn)** The method as defined in Claim 25, wherein each said ID is a user ID.

**27.** **(Withdrawn)** The method as defined in Claim 25, wherein when the verifying verifies that the first assembly is privileged to access the second assembly:

loading the native code with a CLR loader in a native code portion that includes an operating system; and

executing the compiled native code in the native code portion, wherein the first assembly accesses the second assembly.

**28. (Withdrawn)** The method as defined in Claim 25, wherein the determining determines that the respective IDs associated with the first and second assemblies are accurate at a runtime for the native code.

**29. (Withdrawn)** The method as defined in Claim 25, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by the first and second assemblies in respective application domains.

**30. (Withdrawn)** The method as defined in Claim 25, wherein:

the compiling is performed by a JIT compiler; and

the native code portion includes a CLR loader to load the compiled native code.

**31. (Withdrawn)** The method as defined in Claim 25, wherein the verifying further comprises permitting the access of the first assembly to the second assembly

when the ID for the first assembly matches the ID for the second assembly based upon a predetermined match criteria for the respective IDs.

**32. (Withdrawn)** The method as defined in Claim 25, wherein when the verifying verifies that the first assembly is not privileged to access the second assembly:

upon the intercepting of the call:

preventing the access by the first assembly to the second assembly; and

outputting an exception.

**33. (Withdrawn)** A computer readable medium including machine readable instructions for implementing the method as defined in claim 25.

**34. (Withdrawn)** An server comprising:

a managed code portion including:

one or more assemblies in respective application domains; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine;

logic configured to:

register each said assembly as a server object;

intercept a call for access by a first said server object to a second said server object; and

provide access control, based upon an ID for at least one of the first and second said server objects, from the first said server object to the second said server object.

**35. (Withdrawn)** The server as defined in Claim 34, wherein the logic is further configured to receive intermediate language code and metadata generated by a language compiler to form the one or more assemblies in respective application domains.

**36. (Withdrawn)** The server as defined in Claim 35, wherein the intermediate language code and metadata generated by the language compiler from one or more files each having a file type and being associated with user code.

**37. (Withdrawn)** The server as defined in Claim 36, wherein:

the one or more files comprise a database management system for an object-oriented database; and

the server further comprises a network interface for communications with an object-oriented database and with a plurality of clients.

**38. (Withdrawn)** The server as defined in Claim 34, wherein the execution engine further comprises:

a JIT complier to compile said assemblies into native code; and

a CLR loader to load the compiled native code for execution in the native code portion.

**39. (Withdrawn)** The server as defined in Claim 38, wherein the access control further comprises:

determining whether respective IDs associated with the first and second server objects are accurate;

when the determining determines an inaccuracy:

compiling, with the JIT complier, at least one of the first and second server objects from an intermediate language code and metadata into native code; and

when the respective IDs associated with the first and second server objects are accurate:

verifying, based upon the ID for at least one of the first and second server objects, whether the first server object is privileged to access the second server object.

**40. (Withdrawn)** The server as defined in Claim 39, wherein each said ID is a user ID.

**41. (Withdrawn)** The server as defined in Claim 39, wherein when the verifying verifies that the first server object is privileged to access the second server object:

loading, with the CLR loader, the native code compiled by the JIT compiler; and

executing the native code compiled by the JIT compiler in the native code portion, whereby the first server object accesses the second server object.

**42. (Withdrawn)** The server as defined in Claim 39, wherein the determining determines that the respective IDs associated with the first and second server objects are accurate at a runtime for the native code.

**43. (Withdrawn)** The server as defined in Claim 39, wherein when the verifying verifies that the first server object is privileged to access the second server object when the ID for the first server object matches the ID for the second server object based upon a predetermined match criteria for the respective IDs.

**44. (Withdrawn)** The server as defined in Claim 39, wherein when the verifying does not verify that the first server object is privileged to access the second server object:

upon the intercepting of the call:

the access by the first server object to the second server object is prevented; and

an exception is output.

**45. (Withdrawn)** An server comprising logic for providing an identity based security access permission model that maps access rights for a specific database to access rights for a server object, wherein:

one or more said server objects are registered assemblies of the server; and

the server compiles the registered assemblies in managed code into native code that is executed by a common language runtime via the server's operating system.

**46. (Withdrawn)** The server as defined in Claim 45, wherein when the identity based security access permission model is a user identity based security access permission model.

**47. (Currently amended)** A server comprising:

a virtual machine, instantiated in managed code to execute with a runtime loader, to execute first and second assemblies of one or more files instantiated in the managed code, each of the first assembly and the second assembly being registered as a server object with the server;

a first module to make a call for access by the first assembly to the second assembly at Just-In-Time (JIT) compilation time;

an intercept module to intercept the call from the first assembly to the second assembly; [[and]]

a second module, based upon [[an]] user identification (ID) for at least one of the first and second assemblies, to determine access privileges of the first assembly to the second assembly; and

a JIT compiler module, based upon a first determination made at the second module that it is unknown whether the call from the first assembly to the second assembly should be permitted, to perform actions comprising:

inserting a runtime stub into the call; and

compiling the first assembly and the second assembly in the managed code into native code for execution as native code, wherein at runtime when the native code of the first assembly and the second assembly is executed at the server, the second module of the server is configured to make, based upon the user ID for each of the first assembly and the second assembly at the runtime, a second determination of whether the call by the first assembly to the second assembly shall be permitted at the runtime.

**48. (Currently amended)** The apparatus as defined in Claim 47, further comprising:

an execution engine, instantiated in a native code, to execute the virtual machine in runtime;

~~a compiler to compile each of said first and second assemblies into native code for execution as native code;~~ and

an operating system in native code to be executed with one or more of the compiled first and second assemblies.